

Per saperne di più

Il Wiki di Nokia dedicato alla piattaforma Python:
<http://wiki.opensource.nokia.com/projects/PyS60>

- Le risorse per sviluppatori di Nokia:
<http://forum.nokia.com>

- Python per Nokia Serie 60:
<http://sourceforge.net/projects/pys60>

- Putools, un ambiente a riga di comando completo e funzionale:
<http://people.csail.mit.edu/kapu/symbian/python.html>

eseguire comandi da tastiera in una finestra di terminale. Fare la stessa cosa in una connessione senza fili con un telefono dà un po' di emozione. Per fare funzionare la connessione bisogna avere a disposizione una porta seriale Bluetooth. Si può verificare qual è la porta corretta navigando nel menu di avvio lungo il percorso Impostazioni, Pannello di controllo, Rete, Dispositivi Bluetooth. In una macchina utilizzata comunemente per sincronizzare il telefono sarà già presente l'icona del telefono nella cartella Periferiche, diversamente occorrerà avviare la procedura di accoppiamento cliccando sul pulsante Aggiungi. La scheda Porte COM mostra tutte le porte seriali emulate associate alla connessione wireless.

Se non c'è già una porta associata al telefono, bisognerà aggiungere una porta in ingresso con un clic sul pulsante Aggiungi. Per iniziare la sessione occorre avviare Hyperterminal – che nel menu di avvio è nella cartella Accessori, Comunicazioni – creando una sessione collegata alla porta COM appena creata. Successivamente si può avviare la console Bluetooth sul telefono.

Usare al meglio la connessione

Dalla finestra di comando si possono digitare istruzioni Python, come `print 2 + 2`, e si possono eseguire

script con l'istruzione `execfile`, per esempio

```
execfile("snake.py").
```

Con una connessione a riga di comando possiamo sperimentare un po' con l'interfaccia e con le funzioni di libreria. Oltre alle librerie di Python 2.2 abbiamo anche dei moduli specifici per il telefono. Un telefono Symbian infatti è capace di multitasking, di gestire diversi thread, ha la possibilità di creare diversi tipi di connessioni di rete, fare fotografie, inviare Sms.

Curiosando nella libreria troviamo diversi ritagli di codice interessante, per esempio come prelevare del testo dal web

```
urllib.urlretrieve(una_url, tempfile)
f = open(tempfile, 'r')
weatherinfo = file.read()
f.close()
```

Come leggere da un socket, in particolare come leggere una riga alla volta da una connessione Bluetooth

```
class socket_stdio:
    def __init__(self, sock):
        self.socket=sock
    def read(self, n=1):
        return self.socket.recv(n)
    def write(self, str):
        return self.socket.send(str.replace('\n', '\r\n'))
    def readline(self, n=None):
        buffer=[]
        while 1:
            ch=self.read(1)
            if ch == '\n' or ch == '\r': # return
                buffer.append('\n')
                self.write('\n')
                break
            if ch == '\177' or ch == '\010': #
                backspace
                self.write("\010 \010") # erase
                character from the screen
                del buffer[-1:] # and from the
                buffer
            else:
                self.write(ch)
                buffer.append(ch)
            if n and len(buffer)>=n:
                break
        return ''.join(buffer)
    def raw_input(self, prompt=""):
        self.write(prompt)
        return self.readline()
```

```
def flush(self):
    pass
```

Il modulo `appuifw` contiene funzioni specifiche per l'interfaccia utente, ecco un esempio minimo di creazione di un'interfaccia utente e una finestra di dialogo.

```
import appuifw
# titolo dell'interfaccia
appuifw.app.title = u"Hello World"
# una finestra di notifica
appuifw.note(u"Hello World!", 'info')
```

Naturalmente si possono manipolare le applicazioni del telefono, per esempio il calendario e la rubrica. Ecco un esempio di codice per gestire la rubrica

```
new_entry = db.add_appointment()
new_entry.set_time(now+2*week, now+2*
week+hour)
new_entry.content='calendar test'
new_entry.location='somewhere'
```

L'interfaccia verso il database degli indirizzi permette di usare istruzioni in SQL. Ecco come possiamo leggere tutti gli eventi in calendario e trasferirli in una lista

```
dbv = e32db.Db_view()
dbv.prepare(self.native_db,
u"SELECT * from events ORDER BY date DESC")
dbv.first_line()
results = []
for i in range(dbv.count_line()):
    dbv.get_line()
    e = SportsDiaryEntry(dbv)
    results.append(e)
    dbv.next_line()
return results
```

Qualcosa di meglio

Dopo un po' di sperimentazione si inizia a desiderare qualcosa di meglio della console standard, per diverse ragioni. L'editing della riga di comando rudimentale, Hyperterminal non è un'applicazione molto pratica, modificare i programmi sul Pc e trasferirli via Bluetooth riempie la casella dei messaggi e crea un proliferare di versioni intermedie e non si possono manipolare i file caricati. Esiste una console mol-

to migliore, che risolve tutti questi problemi. Il software si può scaricare dall'indirizzo Internet <http://people.csail.mit.edu/kapu/symbian/python.html> e consiste di una console Bluetooth modificata e un'applicazione client scritta in Python. Dato che Python è usato anche sul computer, occorre installarlo per sfruttare l'interfaccia. Le istruzioni per configurare putools sono molto puntuali e tutti i pacchetti necessari si installano senza richiedere attenzioni particolari.

Dopo l'installazione di putools si ha a disposizione una riga di comando un po' più potente, con semplici comandi come *cd*, *pwd*, *ls* e *rm* per maneggiare gli script installati, *man* per avere informazioni sui comandi, *run* per eseguire codice e i file.

Il comando più utile è *sync*, che permette di sincronizzare una directory sul Pc con una sul telefono, in questo modo si può usare l'editor preferito per creare codice Python, per esempio Idle che viene



Una finestra di terminale aperta sul server. Ma stavolta il server è un telefono, il client un Pc e la connessione è Bluetooth.

distribuito con Python, e trasferire i file sul telefono molto semplicemente. Si possono anche visualizzare immagini con il comando *view* e si può catturare lo schermo

del telefono con il comando *snap*. In definitiva, si ha a disposizione una frazione consistente del controllo e della potenza di un Sdk completo di emulatore con in più la velocità del ciclo di sviluppo tipica di un interprete e l'aiuto di un linguaggio semplice da leggere e da scrivere con una sintassi logica e facile da capire.

Per fare ulteriori passi avanti, se ci si vuole dedicare seriamente allo sviluppo su Symbian, si può scegliere un ambiente di sviluppo Java o C++ e imbarcarsi nella curva di apprendimento di un ambiente completo, ma sicuramente l'ambiente Python resta una soluzione valida almeno per la creazione di prototipi e per esperimenti, ma si può considerare Python anche al di là del prototipo, dato che è possibile creare il pacchetto di installazione di una soluzione includendo anche l'ambiente run time del linguaggio. •